D3.1: Initial Report on Interactive Editing

Jesús González-Rubio, Hieu Hoang, Philipp Koehn, Herve Saint-Amand

Distribution: Public

CASMACAT
Cognitive Analysis and Statistical Methods
for Advanced Computer Aided Translation

ICT Project 287576 Deliverable D3.1

Project funded by the European Community
under the Seventh Framework Programme for
Research and Technological Development.

Project ref no.	ICT-287576
Project acronym	CASMACAT
Project full title	Cognitive Analysis and Statistical Methods for Advanced Computer Aided Translation
Instrument	STREP
Thematic Priority	ICT-2011.4.2 Language Technologies
Start date / duration	01 November 2011 / 36 Months

Distribution	Public
Contractual date of delivery	October 31, 2012
Actual date of delivery	November 11, 2013
Date of last update	November 11, 2013
Deliverable number	D3.1
Deliverable title	Initial Report on Interactive Editing
Type	Report
Status & version	Draft
Number of pages	39
Contributing WP(s)	WP7
WP / Task responsible	UEDIN, UPVLC, CBS, CS
Other contributors	
Internal reviewer	
Author(s)	Jesús González-Rubio, Hieu Hoang, Philipp Koehn, Herve Saint-Amand
EC project officer	Kimmo Rossi
Keywords	

The partners in CASMACAT are:

University of Edinburgh (UEDIN)
Copenhagen Business School (CBS)
Universitat Politècnica de València (UPVLC)
Celer Soluciones (CS)

For copies of reports, updates on project activities and other CASMACAT related information, contact:

The CASMACAT Project Co-ordinator
Philipp Koehn, University of Edinburgh
10 Crichton Street, Edinburgh, EH8 9AB, United Kingdom
pkoehn@inf.ed.ac.uk
Phone +44 (131) 650-8287 - Fax +44 (131) 650-6626

Copies of reports and other material can also be accessed via the project's homepage: http://www.casmacat.eu/

Executive Summary

This deliverable reports on work carried out in work package 3 in the first year of the CASMACAT project. In year 1, the addressed tasks were: Sentence-level and word-level confidence measures (Task 3.1 and 3.2), which resulted in two publications, one each by UEDIN and UPVLC; integration of rules from translation memory fuzzy matches (Task 3.3), and visualization of word alignment in the CASMACAT editor (Task 3.4). The latter two tasks were completed in year 1 and may only be refined slightly in the coming years.

Contents

1 Sentence-level Estimate of Post-editing Work Effort (Task 3.1)

The goal of this task is to develop techniques to estimate the amount of effort required to fix automatic translations. The work for this task is based on our earlier research in sentence-level confidence measures (González-Rubio et al., 2010a, 2011, 2012). Specifically, we have focused on two different issues: how to effectively combine subsequence-level features into sentence-level features, and how to select the most adequate subset of features to train reliable prediction models. To test the techniques developed, we participated on the quality estimation task held on the 2012 workshop on statistical machine translation (WMT 2012) (Callison-Burch et al., 2012). The article (González-Rubio et al., 2012) describing our participation can be found on Attachment A.

In (Buck, 2012) we explore the use of black-box features for Quality Estimation. As above, this work was carried out in the context of the WMT 2012 quality estimation task. Here, we focus on finding new, informative features that can be computed without knowledge of the inner workings of the MT system that produced the translation. The proposed features can be split into three categories: (1) indicatiors of simple phenomena like quotation marks and named entites that point out structurally complex segments; (2) features that rate the choice of words in the target sentence using e.g. a log-linear model or neural networks; (3) features that model the distance between the source sentence and the training corpus. The full paper can be found in Attachment B.

Next, we include a further description of our work on feature reduction and the integration of these sentence-level quality estimation measures into the CASMACAT workbench.

1.1 Introduction

Sentence-level quality estimation (QE) is typically addressed as a regression problem (Quirk, 2004; Blatz et al., 2004; Specia et al., 2009b). Given a translation generated by an MT system (and potentially other additional sources of information) a set of features is extracted. Then, a model trained using a particular machine learning algorithm is employed to compute the quality score from these features. Most QE works consider a fixed set of features and study the performance of different learning algorithms on those features. However, feature sets tend to be highly redundant, i.e. there is high multicollinearity between the features, and some of the features may even be irrelevant to predict the quality score. Moreover, a set of translations labelled with their "true" quality score is required to train the learning model. Since this labelling process is usually done manually, training sets rarely contain enough labelled samples to accurately train the model. By removing irrelevant and redundant features from the data, dimensionality reduction (DR) methods potentially improve the performance of learning models by alleviating the effect of the "curse" of dimensionality, enhancing generalization capability of the model, and speeding up the learning process. Additionally, DR also help the researchers to acquire better understanding about their data by telling them which are the important features and how they are related with each other. Despite these potential performance improvements, works on QE usually put little attention on DR. For example, approximately half of the participants in the quality estimation task of the 2012 workshop on statistical machine translation (Callison-Burch et al., 2012) do not apply DR, and even those participants that use DR only implement simple feature selection methods.

We propose two novel DR methods for QE based on partial least squares regression (PLSR) (Wold, 1966). We consider both a DR method that selects a subset of the original features (feature selection method), and a method that projects the original data into a space of fewer dimensions (feature extraction method). Despite being usually more complex, feature extraction methods

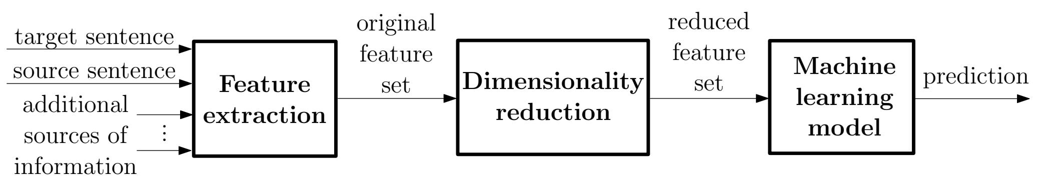


Figure 1: Dataflow of a quality estimation system with feature reduction.

have a potential advantage over feature selection: they can generate new features that summarize the "information" contained in all original features. In contrast, the information contained in the features discarded by a feature selection method is inevitably lost. The proposed methods are compared to other DR methods previously used in the literature: methods based on statistical multivariate analysis such as PCA (Pearson, 1901) and PLSR regressors selection (Specia et al., 2009b), and heuristic wrapper selection methods (Kohavi and John, 1997). Moreover, we study how these DR methods affect the performance of different learning models.

The performance of each DR method was evaluated by the prediction accuracy of the models trained in the corresponding reduced feature sets. Figure 1 shows a scheme of the process followed to obtain a quality score from a given translation. First, from the translation, and additional information sources, we compute a (possibly high-dimensional and highly-redundant) set of features that represent the translation. Then, we apply a DR method to obtain a reduced feature set that still contains the relevant information present in the original feature set. Finally, we use a trained learning model to predict the quality score of the translation from this reduced feature set. To assure an accurate comparison between the different DR methods, identical pipelines were used to train the models. By providing a detailed description and a systematic evaluation of these DR methods, we give the reader various criteria for deciding which method to use for a given task.

1.2 Dimensionality Reduction Methods

Next, we describe the different DR methods tested in the experimentation. For a more clear presentation, we distinguish between heuristic methods and methods derived from statistical multivariate analysis.

1.2.1 Heuristic feature selection methods

We consider heuristic wrapper (Kohavi and John, 1997) methods to address the problem of feature selection. In the wrapper methodology, the learning model is considered a perfect black box. In its most general formulation, this methodology consists in using the prediction accuracy of a given learning model to assess the relative usefulness of subsets of features. In practice, the different wrapper methods are defined by the search strategy implemented to explore the space of possible subsets. An exhaustive search can conceivably be performed, if the number of features is not too large. But, the problem is known to be NP-hard (Amaldi and Kann, 1998) and the search quickly becomes computationally intractable.

In the experimentation, we tested two search strategies that define two different heuristic feature selection methods: ranking of feature selection, and greedy forward feature selection. Since the computational complexity of these simple methods depends on the complexity of the chosen learning model, we use symbol $\zeta(n, m)$ to denote the time complexity to train the actual learning model on n samples of m-dimensional feature vectors.

Rank of feature
Rank of feature selection (RFS) generates subsets of features by selecting the top-scoring features

according to the prediction accuracy of a QE system trained solely with that feature (González-Rubio et al., 2012). RFS is typically used as a baseline selection mechanism because of its simplicity, scalability and (somewhat) good empirical success (Guyon and Elisseeff, 2003). The computational complexity of RFS to generate the first reduced feature set is given by $O(m \cdot \zeta(n, 1))$; once the scores for the features are computed, we can generate reduced groups of different sizes with no further calculations.

Since RFS selects the features according to their individual prediction performance, we expect to obtain subsets of features that also provide good prediction performance. However, RFS does not take into account the correlations that may exist between the different features, thus, these subsets will surely contain a large number of redundant features.

Greedy forward
Greedy forward selection (Kohavi and John, 1997; Avramidis, 2012) (GFS) incrementally creates subsets of features by selecting at each iteration the feature that, when added to the current set, yields the learned model that performs best. In contrast to RFS, GFS recomputes the importance of each feature at each step having into account the current subset of features. Thus, the computational complexity of GFS to compute a reduced set of size r is approximately in $O(r \cdot m \cdot \zeta(n, r))$.

Since GFS selects at each step the feature that improves most the QE model performance, we expect to obtain subsets with lower redundancy in comparison to RFS. However, it requires to re-compute the contribution of each feature to the QE model at each step, which penalizes GFS complexity.

1.2.2 Statistical multivariable analysis

Statistical multivariable analysis is a generic term for any statistical technique concerned with analyzing and understanding data in high dimensions (Anderson, 1958). In particular, we focus on statistical techniques to partition the variability of the data into components attributable to different sources of variation. In this work, we consider two of these techniques: principal component analysis (PCA) and partial least squares regression (PLSR). Given a number of dimensions r, both PCA and PLSR compute a transformation of the original data space into an orthogonal r-dimensional space. However, they differ in the criteria followed to compute this transformation.

The main advantage of these methods stems in the orthogonality of the output space; which means that the transformed features will be linearly independent by construction. Therefore, using these transformations we obtain reduced feature sets with almost no redundant information. Moreover, statistical multivariate methods are mathematically well-founded and independent of the choosen learning model.

Principal component analysis
Principal component analysis (Pearson, 1901) (PCA) defines a transformation of the original data into a new space of features, known as principal components. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint of being uncorrelated with the preceding components. Therefore, each of these principal components represent one of the individual latent factors that actually govern the variability of the data.

PCA projection
The principal components are linearly independent, and each of them accounts for the maximum variability in $\mathbf{X}$ not explained by previous components, thus we follow (González-Rubio et al.,

2012) and select the first r components to create the reduced feature sets. Since each of these components is a linear combination of the original features, thus, this is a feature extraction method. In the experiments, we use PCA projection (PCA-P) to denote this DR approach.

The complexity of PCA-P to compute a reduced set of size r is given by the complexity of the NIPALS algorithm: $O(r \cdot m \cdot n)$. Note that in contrast to the previously presented heuristic methods, the cost of PCA-P does not depend on the complexity of the chosen learning model.

Partial least squares regression

PCA generates sets of orthogonal features where each feature explains the variability of the data $\mathbf{X}$ in one principal direction. However, this transformation of the data ignores the scores $\mathbf{y}$ to be predicted. Thus, although the features generated by PCA-P contain almost no redundancy, they do not necessarily have to be the best-performing set of features. Partial least squares regression (Wold, 1966) (PLSR) is an alternative to PCA that takes into account $\mathbf{y}$ when computing the orthogonal transformation of the data. To do that, PLSR searches for an orthogonal set of latent variables that not only explain the variation in the data (as PCA does), but the variation in the data that best describes the values to be predicted. Mathematically, PLSR can be considered as consisting of two PCA-like transformations (for $\mathbf{X}$ and $\mathbf{y}$ individually) and an inner linear relation $\mathbf{B}$ between them.

Since PLSR is a much more sophisticated model than PCA, different elements of the PLSR model can be used to obtain reduced feature sets. In addtion to the regressors-based selection method previously describe in (Specia et al., 2009b), we propose one new feature selection method: variance importance in projection and one new feature extraction method: PLSR projection. Similarly to PCA-P, the computational complexity of the three PLSR-based DR methods is also given by the complexity of the NIPALS algorithm: $O(r \cdot m \cdot n)$.

Feature importance in regression

Regressor values in a linear relation denote the expected value increment of quality score $\hat{y}$ by unitary increment of feature x_i, i.e., they denote the importance of each feature in the regression. However, due to the usually different scale of the features, these values cannot be directly compared. To do that, first data need to be standarized by subtracting the feature mean form the raw data values and dividing the difference by the standard deviation. By doing that, features become adimensional and regressor values are directly comparable. Thus, we can create subsets of features by selecting them in descending regressor absolute value in $\mathbf{B}$ as done in (Specia et al., 2009b). This method is labelled FIR in the experiments.

Variance importance in projection

After training the PLSR model, we can compute the variance importance in projection (Chong and Jun, 2005) (VIP) of the features. VIP is a score that evaluates the importance of the each feature to find the r latent variables. Therefore, similarly as done for RFS in Section 1.2.1, we propose to select subsets of top-scoring features according to their VIP.

PLSR projection

Finally, under the same motivation as in PCA-P, we propose to obtain a reduced feature set by selecting the first r latent variables. However, in contrast to PCA that obtains the eigenvectors of the covariance matrix ($\mathbf{X}^T\mathbf{X}$), the computation of the eigenvectors in PLSR includes information on the data to be predicted ($\mathbf{X}^T\mathbf{y}\mathbf{y}^T\mathbf{X}$). Thus, a better performance is to be expected. This feature extraction method is labelled PLS-P in the experiments.

1.3 Experiments

1.3.1 Experimental setup

We computed quality scores for translations of the English-Spanish news evaluation data used in the shared QE task featured at the 2012 workshop on statistical MT (Callison-Burch et al., 2012). Evaluation data contains translations of news texts: 1832 sentences for training and 422 sentences for test. Each translation was manually scored by several professional translators in terms of post-editing effort between 1.0 (the translation must be translated from scratch) and 5.0 (the translation requires little to no editing). The final quality score of each translation (a real number in the range $[1, 5]$) is the average of the scores given by the different experts.

We extracted the 480 features described in (González-Rubio et al., 2012) for each of the automatic translations in the evaluation data. As a result, we obtained a training set and a test set of 480-dimensional real vectors with 1832 and 422 samples respectively. All features were standardized by subtracting the feature mean from the raw values, and dividing the difference by the corresponding standard deviation.

Then, we carried out an exhaustive experimentation to test the different DR methods described in Section 1.2. We tested all six DR methods in a series of two-step experiments as depicted in Figure 1. Since we did not knew the dimension of the optimal reduced feature set, each experiment involved several trains of the model with reduced feature sets of different sizes. To tune the meta-parameters of the models, we performed a cross-validation training with ten randomly-chosen data splits.

We evaluated each DR method by the prediction accuracy of the regression model trained on the corresponding reduced feature sets. The performance of a regression model is usually measured by the average error of its predictions $\hat{\mathbf{y}} = \{\hat{y}_1, \ldots, \hat{y}_n\}$ with respect to the actual scores $\mathbf{y} = \{y_1, \ldots, y_n\}$ of the sentences. Specifically, we compute the root mean squared prediction error (RMSPE):

$$\text{RMSPE}(\mathbf{y}, \hat{\mathbf{y}}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{1}$$

where n is the number of test samples. RMSPE quantifies the average deviation of the estimation with respect to the expected score. I.e. the lower the value, the better the performance of the learning model.

1.3.2 Results

Figure 2 displays the cross-validation RMSPE obtained by an support vector machine trained on reduced feature sets generated by the different DR methods presented in the previous section.

The results of the four feature selection methods were very close, and all of them outperformed the baseline model trained in the original 480-dimensional data (0.87 RMSPE). The main differences were due to the number of features required by each model to reach its top performance. Rank of feature selection (RFS) was the method that required more features to converge. Since RFS does not take into account possible correlations between the features, the reduced feature sets generated by this method were highly-redundant; which explains the large number of features it needed to stabilize. In contrast, greedy forward selection (GFS) obtained great improvements with few features. However, it has a higher complexity than other methods which complicates its practical deployment; reason why we carried out experiments only up to 30 features. Finally, both feature selection methods based on PLSR required an intermediate number of features to converge. Results for feature importance in regression selection (FIR)

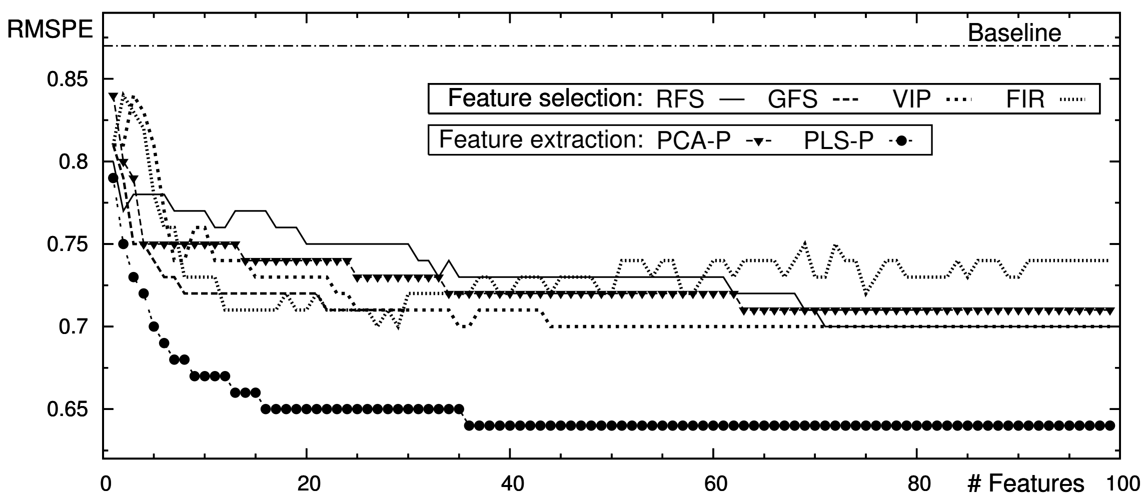


Figure 2: Cross-validation SVMs prediction results on training for different DR methods as a function of the size of the reduced feature set. In comparison, the baseline SVM trained on the 480 original features scored 0.87 RMSPE. PLS-P results were statistically better than the rest.

improved up to ∼ 30 features, then its performance slightly deteriorated as more features were selected. This behavior seems to indicate a tendency of FIR feature sets to over-fitting. Finally, variance importance in projection (VIP) steadily improved performance up to ∼ 40 features where it reached a plateau.

Regarding feature extraction methods, they exhibited important differences in performance. PCA projection (PCA-P) obtained similar results as the four feature selection methods. Its results improved up to ∼ 60 features, and remained stable as the number of features increased. In contrast, PLSR projection (PLS-P) obtained much better results consistently outperforming PCA-P and all feature selection methods. Moreover, these performance differences were significant with a probability of improvement of 95% according to a pair-wise bootstrap analysis (Bisani and Ney, 2004). Additionally, PLS-P only required ∼ 35 features to reach its top performance.

1.3.3 Conclusions

The key results of the experimentation are as follows. First, DR methods that take into account the values to be predicted obtain better results, as exemplified in the great difference between PCA-P and PLS-P. Second, feature extraction methods seem to be more efficient than feature selection methods. Particularly PLS-P that was the best-performing method for reduced feature sets of all sizes. This fact indicates that PLS-P generate more “information-dense” features that constitute a better summary the original high-dimensional feature set. Even a simple method such as PCA-P, that ignores the values to be predicted, obtained comparable results to the best-performing feature selection methods. Finally, the performance-wise ranking of the different DR methods is, to a great extent, independent of the chosen learning model.

One of the proposed DR methods, PLS-P, can be seen as a summary of the conclusions: a feature extraction method based on multivariate analysis that takes into account the values to be predicted to perform the reduction. Thus, it consistently obtained the best results in the experimentation.

1.4 Integration into the CASMACAT Workbench

At the point of the writing of this report, only simple confidence measure based on Model 1 lexicon probabilities (Brown et al., 1993b), have been integrated into a development version of the CASMACAT workbench. In the next months, we plan to implement it, and several other more sophisticated measures, in the official workbench so they can be tested in the next field trial.

2 Word-level Confidence Measures (Task 3.2)

The goal of this task is to develop techiques to determine the reliability of translated words so that those parts of the sentence most likely to require editing are highlighted. The work carried out for this task has focused on the application of word-level confidence measures into a IMT framework as done in our previous research (González-Rubio et al., 2010b). In that article we propose to introduce confidence measures as a new source of information available to the users of an IMT system. We hypothesize that this new information will focus the attention of the human user on those words more likely to be incorrectly translated, thus increasing his translation productivity. The main challenge was to design a word confidence measure that do not hinders the user interaction with the IMT system for which response time is a key issue. In other words, the chosen confidence measure must be as accurate as possible, but also must be very fast to compute so it does not interfere with the interaction process.

We implement a word confidence measure based on the Model 1 (Brown et al., 1993b), similar to the one described in (Blatz et al., 2004; Ueffing and Ney, 2005). We choose this confidence measure because it relies only on the source sentence and the proposed extension, and not on an N-best list or an additional confidence estimation layer as many other word confidence measures do (Blatz et al., 2004). Thus, it can be calculated very fast during search, which, as we have said above, is crucial given the time constraints of interactive systems. Moreover, its accuracy classifying words is similar to that of other word confidence measures as the results presented in (Blatz et al., 2004; Sanchis et al., 2007) show. We modify that confidence measure by replacing the average by the maximal lexicon probability, because work by Ueffing and Ney (2005) showed that the average is dominated by this maximum. Formally, given a source language sentence $\mathbf{x} = x_0 \dots x_j \dots x_J$, the confidence $c(y_i)$ of the words in the corresponding translation $\mathbf{y} = y_1 \dots y_i \dots y_I$ is computed as:

$$c(y_i) = \max_{0 \leq j \leq J} P(y_i | x_j) \ , \tag{2}$$

where $P(y_i|x_j)$ is the Model 1 lexicon probability, x_0 is the empty source word and J is the number of words in the source sentence. After computing the confidence value, each word is classified as either correct or incorrect, depending on whether its confidence value or not a classification threshold ρ.

Empirical results in a simulated setting showed that the confidence measure provided helpful information to locate incorrectly translated words. Thus, it may help the user to ease his interaction with the system. Additionally, the use of word confidence classifiers allowed to modify the behavior of the IMT system so that it ranges between the results of a fully automatic SMT system (when all words were considered correct) and the results of a conventional IMT system (when all words were considered incorrect). In between these two extremes, important reductions in user effort were obtained while retaining a high translation quality.

Next, we describe the integration of this confidence measure in the CASMACAT workbench and the future research lines that will be explored.

2.1 Integration into the CASMACAT Workbench

We have integrated the use of word-level confidence measures into an internal development version of the CASMACAT workbench. We use the Model 1 confidence measure described above to highlight in a different color those words in the translation that are more likely to be incorrectly translated. The idea is that this information will aid the user to detect errors in the translations. Hence, easing his interaction with the system and hopefully improving his translation productivity. In the next months, we plan to implement word confidence estimation into the official CASMACAT workbench. The objective is to use the field trials that will be carried out during the second year of the project to measure the influence of word-highlighting in the work-flow of professional translators, and to verify the potential productivity improvements obtained in previous laboratory experiments (González-Rubio et al., 2010b).

2.2 Future work on word level confidence estimation

In the future we will continue to investigate the integration of word- and sentence-level confidence scores by using sequence models such as conditional random fields as proposed in (Bach et al., 2011). This also includes the computation of additional features, many of which are computed on a sub-sentence level. In particular we aim to include more linguistic information and lexical choice models as well as structural properties of the search graph.

By considering the alignment between the source and the proposed translation we can map later edits back onto the source sentence. We will use this to look for structures that make translation difficult.

Another interesting approach is the prediction of confidence on a finer grained level than words. In many cases, e.g. due to inflection, lexical choices only differ in the last few letters. We will evaluate to what extend this information can be useful in the IMT setting.

We will also explore the connection between automatic MT evaluation scores and confidence estimates in greater detail. This is important due to the practical issue of bootstrapping a confidence estimation system without the appropiate training data, i.e. annotations produced by humans or post-edited segments. In order to produce a meaningful score for a new translation system where such data is absent we can produce all our predictive features but we lack the weights for a combination. Prediction of scores intended for automatic MT evaluation has been used to circumvent this problem, see for example (Specia et al., 2009a), but it remains unclear how this can be turned into a practical solution that yields the expected accuracy.

3 Rules from Translation Memory (Task 3.3)

The goal of this task is to generate translation rules from fuzzy matches from a translation memory, and use them in our statistical machine translation system. The work carried out for this task is based on our earlier research in this area (Koehn and Senellart, 2010a). Here, we refined the method and integrated it into the Moses decoder (Koehn et al., 2007).

3.1 Background

Computer aided translation tools have a well established function called translation memory. The basic idea behind this functionality is very similar to the underlying principle of statistical machine translation. Both re-use a database of prior translations to aid the human translator in her work.

Current statistical machine translation methods use the parallel corpus to extract general rules for translating words, short chunks of texts (phrases) or grammatical mappings (as in tree-based models). A rule is generally preferred if it can be found several times in the corpus.

Translation memory, in contrast, looks for a single full-sentence match in stored translations. If an exact match cannot be found, then a so-called fuzzy match is retrieved, i.e., a sentence that most closely resembles the input sentence currently under consideration. The fuzzy match score used to score fuzzy matches is the typically the Levenshtein distance, a string edit distance that counts the number of insertions, deletions and substitutions, normalized by the input sentence length.

Older work in empirical machine translation that precedes statistical machine translation, so-called example-based machine translation, was also motivated by the maximum sentence-match principle. But in addition to a translation memory look, it also consisted of a so-called recombination phrase where the mismatching part is replaced with other matches.

Viewed from a statistical machine translation perspective, the goal of translation memory is to find a very large translation rule, that combines and competes with other more general rule to produce the most probably translation. Our goal is to add statistical machine translation as additional assistance for human translator. One possible user scenario is to offer a human translator a fuzzy match from a TM, or an SMT translation, or both. What to show may be decided by an automatic classifier (Simard and Isabelle, 2009; Soricut and Echihabi, 2010; He et al., 2010) or may be based on fuzzy match score or SMT confidence measures (Specia et al., 2009c). Here, instead, we integrate the fuzzy match into the statistical machine translation model.

Recent work has explored similar strategies. Motivated by work in EBMT, Smith and Clark (2009); Zhechev and van Genabith (2010) use syntactic information to align TM sentences. Then they create an XML frame to be passed to Moses. Both show weaker performance (on different data sets) than we report here. Smith and Clark (2009) never overcomes the SMT baseline. Zhechev and van Genabith (2010) only beat the SMT system in the 90-99% fuzzy match range.

The work by Biçici and Dymetman (2008) is closer to our approach: They align the TM sentences using GIZA++ posterior probabilities to indentify the mismatch in the target and add one non-contiguous phrase rule to their phrase-based decoder. They show significant improvements over both SMT and TM baselines, but their SMT seems to perform rather badly — it is outperformed by raw TM matches even in the 74-85% fuzzy match range.

3.2 Method

What we describe below is based on our earlier work (Koehn and Senellart, 2010a). This work also considers the construction of XML frames, but found this to be a inferior mechanism. We discard it in favor of the very large rule approach. In short, we reformat the fuzzy match as a hierarchical phrase rule and use it in the decoder alongside other rules.

3.2.1 Fuzzy Matching

The first processing step is to retrieve the best match from the TM. Such fuzzy matches are measured by a fuzzy match score, and the task is to find the best sentence pair in the TM under this score.

There are several different implementation of the fuzzy match score, and commercial products typically do not disclose their exact formula. However, most are based on the string edit distance, i.e., the number of deletions, insertions, and substitutions needed to edit the input sentence to the TM sentence.

Our implementation of the fuzzy match score (Koehn and Senellart, 2010b) uses word-based string edit distance, and uses letter string edit distance as a tie breaker. We define the fuzzy match score as:

$$\text{FMS} = 1 - \frac{\text{edit-distance(source, tm-source)}}{\max(|\text{source}|, |\text{tm-source}|)} \tag{3}$$

3.2.2 Hierarchical Phrase Rules

First some background on hierarchical translation rules. If we have the following sentence pair with monotone word alignment

- *the big fish*
- *les gros poissons*

we can create phrase-based translation rules from the following phrase pairs:

- (*the* ; *les*)
- (*the big* ; *les gros*)
- (*the big fish* ; *les gros poissons*)
- (*big* ; *gros*)
- (*big fish* ; *gros poissons*)
- (*fish* ; *poissons*)

Hierarchical phrase-based rules are constructed by removing a smaller phrase pairs from a larger phrase pairs. For instance, by taking the phrase pair:

(*the big fish* ; *les gros poissons*)

and removing the phrase pair

(*big* ; *gros*)

we create the rule

(*the* X *fish* ; *les* X *poissons*)

The symbol X is called a non-terminal, since the translation rule is viewed as a synchronous context-free grammar rule. In essence it is a place-holder for recursively nested sub-phrases.

Hierarchical rules require a different decoding algorithm than phrase-based rules. The algorithm is typically drawn from syntactic parsing methods, but otherwise use a very similar training, tuning, and testing pipeline as traditional phrase-based models (Hoang et al., 2009).

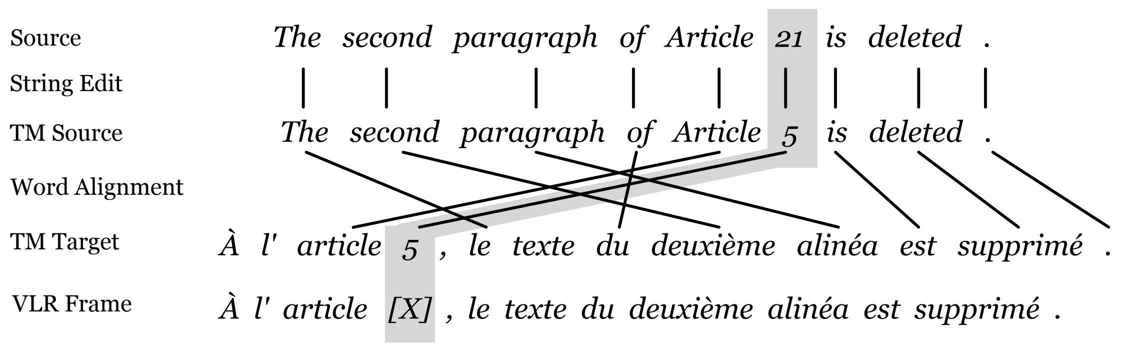

Source
String Edit
TM Source
Word Alignment
TM Target
VLR Frame
The second paragraph of Article 21 is deleted .
The second paragraph of Article 5 is deleted .
À l' article 5 , le texte du deuxième alinéa est supprimé .
À l' article [X] , le texte du deuxième alinéa est supprimé .

Figure 3: **Construction of the Very Large Rule:** The mismatched source word is tracked to the TM target word.

3.2.3 Very Large Rules

Let us now illustrate the process (see also Figure 3) how a fuzzy match is converted into a phrase-based rule. Let us say that the following input sentence needs to be translated:

The second paragraph of Article 21 is deleted .

The TM does not contain this input sentence, but it contains something very similar:

The second paragraph of Article 5 is deleted .

In the TM, this sentence is translated as:

À l' article 5 , le texte du deuxiéme alinéa est supprimé .

The mismatch between our input sentence and the TM source sentence is the word *21* or *5*, respectively. By letting the SMT system translate the true source word *21*, but otherwise trusting the target side of the TM match, we construct the following Very Large Rule:

(*The second paragraph of Article* [X] *is deleted .* ;
À l' article [X] *, le texte du deuxiéme alinéa est supprimé .*)

This Very Large Rule consists of specified translations (e.g., *À l' article*) and a non-terminal that marks the mismatch. When using this rule, the decoder translates the mismatch by consulting a rule table, and search for the best translation according to various scoring functions including a language model. The decode is also free to ignore the Very Large Rule derived from the fuzzy match. In addition, this scheme allows for the extraction of multiple rules that compete against each other.

In practice, hierarchical models do not use such large rules (and keep in mind, this particular rule is drawn from a relatively short sentence, thus containing few words and only one non-terminal). But this is purely due to scaling issues and concerns about the size of the rule table. The issues may be resolved. In fact, Lopez (2007) presented a method to compute very large translation rules on the fly for hierarchical models. While these rules were limited to two non-terminals, they could contain any number of words.

3.2.4 Construction of Very Large Rules

The mismatch between the input sentence and the TM source sentence is easy to detect. As with our fuzzy match metric, we compute the string edit distance between the two, which detects the inserted, deleted and substituted words. A harder problem is to determine which target target words are affected by the mismatch.

For this, we need a word alignment between the source words and the target words in the TM sentence. Word alignment is a standard problem in SMT, and many algorithms have been proposed and implemented. Most commonly used is the implementation of the IBM Models (Brown et al., 1993a) in the toolkit GIZA++ (Och et al., 1999), combined with symmetrization heuristics. This toolkit is also used by the Moses SMT training pipeline. So, if we build an SMT system on the TM data, then the word alignment falls out as a by-product.

We construct the Very Large Rule by subtraction. All of the TM target sentence is passed as a specified translation to the SMT decoder, except for the subtraction of the mismatch. The TM target word aligned to the mismatched source word is not part of a specified translation, and instead the source word is inserted in their place.

A mismatch may consist of a sequence of multiple words in source, TM source, or TM target. We treat such a block the same way we treat single words: they are removed as a block from the TM target and the source block is inserted. There may be multiple non-neighboring mismatched sequences. We treat each separately and perform the subtraction process for each.

There are a number of special cases stemming from mismatched words that are unaligned, aligned at distant points in the target sentences, or completely dropped, with the complementary case of additional words in the TM sentence with no correspondence to words in the input sentence. See (Koehn and Senellart, 2010a) for a detailed discussion.

3.2.5 Results

See Figure 4 for a comparison of straight statistical machine translation, translation memory, and the Very Large Rule approach. Numbers are reported on a English–French test set drawn from the publicly available JRC-Acquis corpus[1] (Acquis). The Acquis corpus is a collection of laws and regulations that apply to all member countries of the European Union. It has more repetitive content than the parallel corpora that are more commonly used in machine translation research. The training corpus consists of over a million sentences with about 25 million words, and the test set of a representative sample of 4,107 sentences. We use the same test set as Koehn et al. (2009).

We are especially interested in the performance of the methods for sentences for which we find highly scoring fuzzy matches, since we do not expect fuzzy matches to be very useful otherwise. See Table 1 for statistics on the subsets based on fuzzy match ranges. The sentences with 100% fuzzy match are much shorter, but otherwise there is no strong correlation between fuzzy match score and sentence length. The table also contains the BLEU scores reported in the graph of Figure 4

3.3 Integration into Moses Decoder

The method described above consists of three steps and was implemented in different programming languages:

[1]http://wt.jrc.it/lt/Acquis/ (Steinberger et al., 2006)

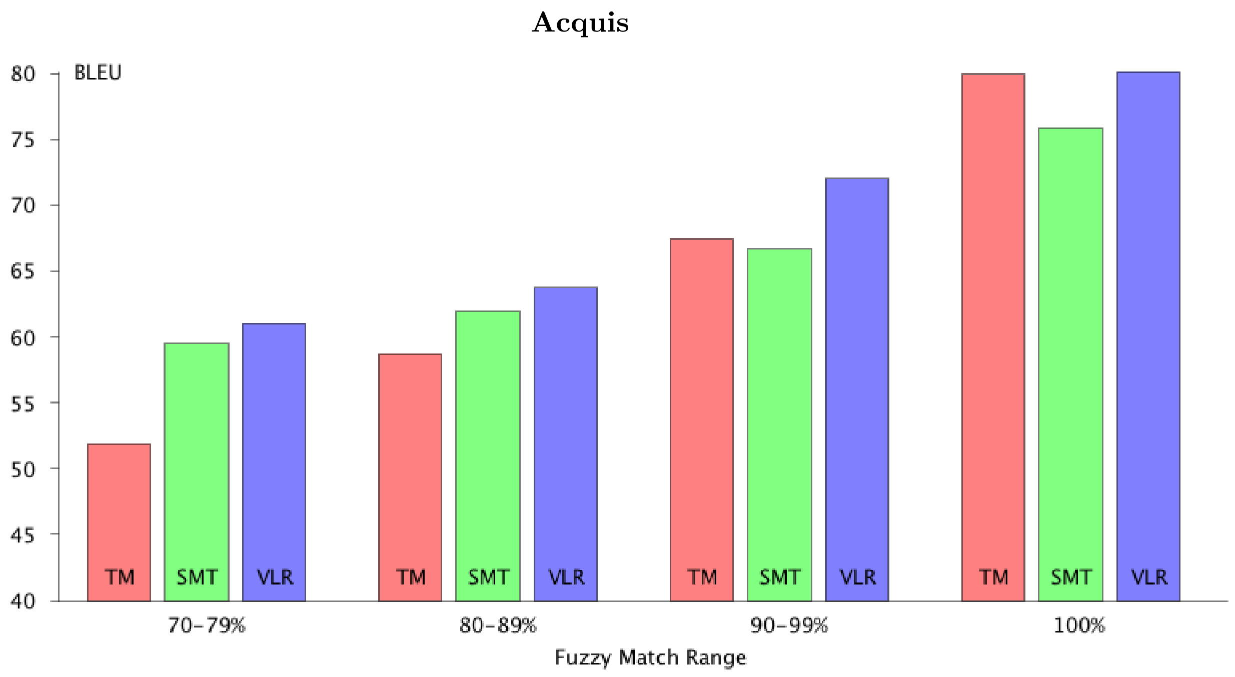


Figure 4: TM matches as very large rules (VLR): Encoding TM match as very large hierarchical grammar rules (VLR) outperforms baseline methods.

	Corpus Statistics			BLEU		
Acquis	Sentences	Words	W/S	SMT	TM	VLR
100%	1395	14,559	10.4	79.9	75.8	80.1
90-99%	433	12,775	29.5	67.4	66.6	72.0
80-89%	154	5,347	34.7	58.7	62.0	63.7
70-79%	250	6,767	27.1	51.9	59.5	61.0

Table 1: Composition of the test subsets based on fuzzy match scores (letter-based string edit distance) and quality of the Very Large Rule approach compared against the baseline statistical machine translation system (SMT) and unmodified translation memory matches (TM).

- a fuzzy matcher (Koehn and Senellart, 2010b) that finds the most similar source sentences (and their translation) in the parallel corpus, implemented in C++
- a rule constructor based on the fuzzy match with its edit path and the word alignment between source and target, implemented in Perl
- a secondary rule table file provided to the Moses decoder

The main disadvantage of this implementation is that the full text to be translation has to be know before the decoder is started up — which makes it unusable for online use.

Hence, we integrated the method as an additional rule table format into the Moses decoder. Moses already supports a number of rule tables. Most are variants of storing a fixed set of rules (phrase-based or grammar-based), with different optimizations in mind (read from a text file, queries towards an on-disk database, stored in memory with and without compression). But one related table format stores the parallel corpus with word alignment in memory and offers a suffix array to enable quick queries. In fact, the suffix array is dynamic, allowing for incremental updates. While the decoder is running, new sentence pairs can be added on the fly (Levenberg et al., 2010). Our method stores the translation memory also in a suffix array but uses it a uniquely different way. Instead of querying the suffix array to mimic the extraction of a general purpose translation model, the fuzzy matching algorithm uses it to find n-gram matches as a filtering step to retrieve sentence matches (Koehn and Senellart, 2010a). Based on the sentence matches, Very Large Rules are constructed. The method also return the probability features that are used by the Moses decoder, namely the probability of target given source, *source given target*, and a constant rule count feature.

When integrating the algorithm into the Moses decoder, we take advantage of the fact that all rule tables are consulted prior to the translation of each sentence. This enables the execution of the fuzzy matching algorithm, retrieving the most similar translation memory matches to create the very large translation rules.

Integrating the fuzzy matching algorithm into the Moses decoder as another variant of the rule table offers several advantages. Firstly, the algorithm can now be used online, without the need to preprocess the full input text. Secondly, using the existing framework allows the fuzzy matching algorithm to make use of the features in the Moses decoder, such as multi-threading and error handling. Lastly, packaging the fuzzy matching algorithm as a rule table makes it very convenient to use: only a parallel corpus with word alignment has to be specified, and the decoder treats it just like any other rule table implementation.

3.4 Integration into CASMACAT Workbench

The Moses decoder is able to pass along with the translation annotation about which words were produced by which rules. This allows us to distinguish between words that were generated by the regular rule table and words that were generated with a very large rule based on a fuzzy match.

We plan to highlight words not generated by fuzzy match rule in the user interface of the CASMACAT workbench. The implementation of this is very similar to the markup with confidence scores. The idea is that words generated by the fuzzy match are more likely to be trusted, but there may be errors when filling in the mismatched part. By directing the human translator's attention to the mismatched part, she is able to correct eventual errors quickly.

At the point of the writing of this report, the integration into the workbench has not yet happened, due to delays with the tool development. We expect this task to be completed within 1–2 months and the new functionality tested in the next field trial.

4 Visualisation of Word Alignment (Task 3.4)

Another of the new methods being developed and investigated as part of the CASMACAT project aiming to assist translators in their work is the highlighting of alignments between source and edited translation. In a nutshell, this consists in visually highlighting on the computer screen the words or phrases that are currently the focus of the translator's attention, both within the source text and within the translation. The goal is to guide the translator's eye towards the parts of the text they are currently focusing on. Figure 5 shows an example.

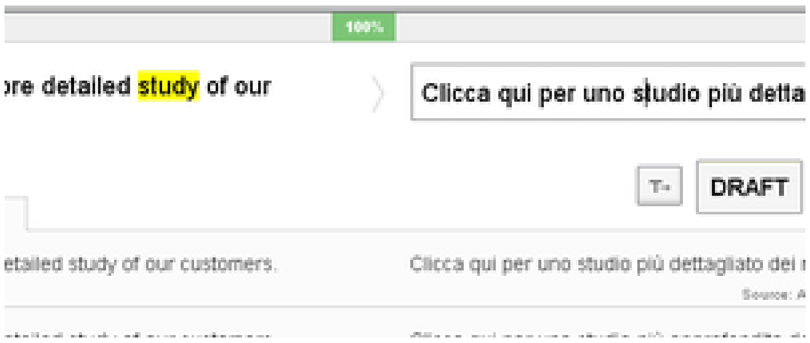

100%
ore detailed study of our
Clicca qui per uno studio più detta
DRAFT
etailed study of our customers.
Clicca qui per uno studio più dettagliato dei
Source: A

Figure 5: An example of source word highlighting: the translator, typing on the right, has their cursor on the Italian word "studio". The corresponding word in the English source, "study", is highlighted on the left.

4.1 Background

The concept of alignment visualisation rests upon a few working hypotheses. These hypotheses constitute the basis of the motivation and the driving design principles for this work. The goal of task 3.4 is to implement software guided by these hypotheses, and integrate them into the CASMACAT workbench. As we then proceed to field trials, where professional translators will use the workbench to produce translations, the usefulness and performance of this feature can then be evaluated, allowing us to confirm, refine, or disprove our working hypotheses.

These hypotheses are as follows.

4.1.1 Focused Attention

As a translator types, edits or reviews their translation, their focus of attention will typically be concentrated onto a single word or phrase at a time. This "window of attention" will slide over the entire text as work progresses, but will not expand to cover the entire text. In other words, when translating a sentence of non-trivial length (i.e. any sentence of more than just a few words), the translator will not concentrate on the entire sentence at once, but rather will work on it one phrase, or even one word at a time.

As they do so, their visual attention will naturally be drawn to the part of the Graphical User Interface where translation input is performed (text box, editor frame, etc.), as that is where the text input and editing operations need to be performed. At the same time, however, they will be trying to keep in mind the word or phrase in the source text that corresponds to the word or phrase they are currently typing. As their visual gaze is diverted from the source text to be focused on the translation they are typing, however, they might lose track of their position in the source.

4.1.2 Highlighting as a Visual Aid

It is hypothesised that changing the colour of the background onto which the text is written, for instance by turning it to a yellow colour, will aid the translator in keep track of their position within the source text, even as their gaze is brought to the translated text by the needs of computer input and text editing.

4.2 Implementation

In this section we give an overview of the approach taken to implement the alignment highlighter.

The realisation of the application is a two-step process. The first is an offline pre-computing step, during which data is gathered, then compiled into a format suitable for efficently handling queries, and finally stored in the backend database. The second step occurs live as the translator is using the application, and consists in continuously sending queries to the backend database.

4.2.1 Pre-computing of alignments

The first step in setting up the alignment highlighting feature is to train the alignment models. This is done offline, ahead of time, and only needs to be done once.

Training Data A large parallel corpus is needed. The ideal corpus is large (hundreds of millions of words), and has a general writing style that matches as closely as possible the style of the documents being translated. In most real-life scenarios, of course, it is impossible to know in advance what style of documents will be translated by the user, so we aim instead to cover as many bases as possible by including text that is not too specific to any domain, or that covers several likely domains.

We chose to use the Europarl corpus because it was readily available in several pairs of the major European languages, it is reasonably large, it contains very well-aligned data, and it has already been used extensively, meaning that our tools have all been well tuned to its contents. The Europarl corpus does however mostly constist of the somewhat idiosyncratic language of parliament debates, being as it is entirely composed of such debates. Still, it is general enough for the corpus to be a very valuable training resource.

This is however only a baseline system. We intend to re-train our models with larger corpora spanning more domain-specific genres, to diversify the covered vocabulary.

Extracted Models From this corpus we extract two distinct models. Both are computed using the GIZA++ software suite, a well proven tool that is commonly used in building statistical machine translation systems.

The first model that we extract from the data is a lexical translation model. In simple terms, this is simply a count, for every word f in the source language, of all possible target language words e that could be used a translations of f, along with associated probabilities. As a rough example, the lexical translation model might tell us, given a French-English parallel corpus, that the French word “livre” translates into English as “book” 65% of the time, as “pound” 15% of the time, and so on.

It is important to note that the corpus being fed to GIZA++ consists of pairs of whole sentences in a plain text format, without any annotation that may indicate which words within a sentence in the source language correspond to which in the target language. The software needs to work out the most probable alignments using word repetitions over sentence pairs.

The second model that we extract from the corpus using GIZA++ is an alignment model. This model is also a set of probabilities inferred from the training data, but in this case the probabilities model not lexical entries but the position of words in one language respective to the other. In simple words, we model how much words tend to move about the sentence when translated from the source language to the target.

There exist many mathematical models of word alingment. The specific type of alignment model that we use is called an HMM alignment model. Whereas the lexical alignment that we extract using GIZA++ corresponds to a standard, state of the art baseline system, an HMM alignment model is considerably simpler, and consequently less accurate, than many other models available to us. As we shall see, however, this choice of a simplistic model is constrained by the limited computational resources available to us in the browser. The HMM model represents a good tradeoff, affording us decent accuracy while being computationally very cheap to compute.

Compiling of extracted data Two sorts of lookups will be necessary as the user interacts with the application. First, when the application is loaded up, the alignment model is fetched from the server and loaded to memory. Since the HMM model records alignment probabilities for a given source text position i given only the alignment used in the preceding position $i - 1$, the model is very compact.

A single file of alignment probabilities from the corpus is therefore pre-computed using GIZA++, converted to JSON format for easy parsing in the browser, gizpped and stored on disk, to be directly read by the Web application on startup.

The second sort of lookup occurs as the user selects a new sentence to translate. At that point, the lexical translation model is loaded for each of the words on the source side of the sentence. Since the source sentence is fixed and not editable by the translator, this model only needs to be loaded once per sentence. Afterwards, no matter what the translator types as a translation, we'll be able to look up in the model the probabilities of the translated words, given the (fixed) source words.

4.2.2 Live updating of the highlighting

The GUI needs to be updated every time the user moves their caret around, selects a new sentence pair to work on, or edits the currently focused translation.

As is usual in web-based applications, this is all programmed in event-driven fashion. Events are attached to the user actions, and trigger a re-evaluation of the highlighter every time the focused translated word changes. Translated word is computed as sentence offset + word token. At every re-evaluation all highlighting is removed, the most probable alignment between the sentences is re-computed, and the source word that corresponds in that alignment to the focused translated word is highlighted.

At present highlighting is implemented as a simple DOM operation that replaces the word to highlight with a `<span>` node having a bright yellow background.

4.2.3 A Note on Tokenization

Tokenization proved to be a somewhat unexpected challenge in this application. In order to be able to find which word the user is currently editing, the character string needs to be tokenized. However, the extracted tokens will then be used as lookups in models that were built on the server side, using a different tokenizer. That tokenizer, the `tokenizer.perl` script that ships with the Europarl corpus, has its own set of heuristics and fixed, manually compiled list of known exception words.

Ideally an exact port of the server-side tokenizer to JavaScript will be necessary. In the meantime the client-side tokenizer was made roughly equivalent to the server-side one, but disparities still do exist, and they will influence the quality of the outcome.

5 Display Multiple Translation Options (Task 3.5)

This task is planned to run throughout the entire time span of the project. Due to delays in the tool development, we scheduled the start of work into year 2.

References

Amaldi, E. and Kann, V. (1998). On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209(1-2):237–260.

Anderson, T. W. (1958). *An Introduction to Multivariate statistical Analysis.* Wiley, New York.

Avramidis, E. (2012). Quality estimation for machine translation output using linguistic analysis and decoding features. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 84–90, Montréal, Canada. Association for Computational Linguistics.

Bach, N., Huang, F., and Al-Onaizan, Y. (2011). Goodness: A method for measuring machine translation confidence. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Techologies*, pages 211–219, Portland, Oregon, USA. Association for Computational Linguistics.

Biçici, E. and Dymetman, M. (2008). Dynamic translation memory: Using statistical machine translation to improve translation memory. In Gelbukh, A. F., editor, *Proceedings of the 9th Internation Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*, volume 4919 of *Lecture Notes in Computer Science*, pages 454–465. Springer Verlag.

Bisani, M. and Ney, H. (2004). Bootstrap estimates for confidence intervals in asr performance evaluation. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 409–412, Montreal.

Blatz, J., Fitzgerald, E., Foster, G., Gandrabur, S., Goutte, C., Kulesza, A., Sanchis, A., and Ueffing, N. (2004). Confidence estimation for machine translation. In *Proceedings of the International Conference on Computational Linguistics*, page 315.

Brown, P. F., Della-Pietra, S. A., Della-Pietra, V. J., and Mercer, R. L. (1993a). The mathematics of statistical machine translation. *Computational Linguistics*, 19(2):263–313.

Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., and Mercer, R. L. (1993b). The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311.

Buck, C. (2012). Black box features for the wmt 2012 quality estimation shared task. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 91–95, Montréal, Canada. Association for Computational Linguistics.

Callison-Burch, C., Koehn, P., Monz, C., Post, M., Soricut, R., and Specia, L. (2012). Findings of the 2012 workshop on statistical machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada. Association for Computational Linguistics.

Chong, I. and Jun, C. (2005). Performance of some variable selection methods when multicollinearity is present. *Chemometrics and Intelligent Laboratory Systems*, 78(1–2):103–112.

González-Rubio, J., Ortiz-Martínez, D., and Casacuberta, F. (2010a). Balancing user effort and translation error in interactive machine translation via confidence measures. In *Proceedings of the conference of the association for computational linguistics*, pages 173–177, Uppsala, Sweden. Association for Computational Linguistics.

González-Rubio, J., Ortiz-Martínez, D., and Casacuberta, F. (2010b). On the use of confidence measures within an interactive-predictive machine translation system. In *Proceedings of 14th Annual Conference of the European Association for Machine Translation.* http://www.mt-archive.info/EAMT-2010-Gonzalez-Rubio.pdf.

González-Rubio, J., Ortiz-Martínez, D., and Casacuberta, F. (2011). An active learning scenario for interactive machine translation. In *Proceedings of the 13th International Conference on Multimodal Interaction*, pages 197–200.

González-Rubio, J., Ortiz-Martínez, D., and Casacuberta, F. (2012). Active learning for interactive machine translation. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 245–254.

González-Rubio, J., Sanchís, A., and Casacuberta, F. (2012). Prhlt submission to the wmt12 quality estimation task. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 104–108, Montréal, Canada. Association for Computational Linguistics.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Machine Learning Research*, 3:1157–1182.

He, Y., Ma, Y., Way, A., and van Genabith, J. (2010). Integrating n-best smt outputs into a tm system. In *Coling 2010: Posters*, pages 374–382, Beijing, China. Coling 2010 Organizing Committee.

Hoang, H., Koehn, P., and Lopez, A. (2009). A unified framework for phrase-based, hierarchical, and syntax-based statistical machine translation. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*, pages 152–159.

Koehn, P., Birch, A., and Steinberger, R. (2009). 462 machine translation systems for europe. In *Proceedings of the Twelfth Machine Translation Summit (MT Summit XII)*. International Association for Machine Translation.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C. J., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Koehn, P. and Senellart, J. (2010a). Convergence of translation memory and statistical machine translation. In *AMTA Workshop on MT Research and the Translation Industry*.

Koehn, P. and Senellart, J. (2010b). Fast approximate string matching with suffix arrays and a* parsing. In *Meeting of the Association for Machine Translation of the Americas (AMTA)*.

Kohavi, R. and John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324.

Levenberg, A., Callison-Burch, C., and Osborne, M. (2010). Stream-based translation models for statistical machine translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 394–402, Los Angeles, California. Association for Computational Linguistics.

Lopez, A. (2007). Hierarchical phrase-based translation with suffix arrays. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 976–985.

Och, F. J., Tillmann, C., and Ney, H. (1999). Improved alignment models for statistical machine translation. In *Proceedings of the Joint Conference of Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP-VLC)*, pages 20–28.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559–572.

Quirk, C. (2004). Training a sentence-level machine translation confidence metric. In *Proceedings of the International Conference on Language Resources and Evaluation*, pages 825–828.

Sanchis, A., Juan, A., and Vidal, E. (2007). Estimation of confidence measures for machine translation. In *Proceedings of the Machine Translation Summit*, pages 407–412.

Simard, M. and Isabelle, P. (2009). Phrase-based machine translation in a computer-assisted translation environment. In *Proceedings of the Twelfth Machine Translation Summit (MT Summit XII)*. International Association for Machine Translation.

Smith, J. and Clark, S. (2009). EBMT for SMT: A new EBMT-SMT hybrid. In Forcada, M. L. and Way, A., editors, *Proceedings of the 3rd International Workshop on Example-Based Machine Translation*, pages 3–10.

Soricut, R. and Echihabi, A. (2010). Trustrank: Inducing trust in automatic translations via ranking. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 612–621, Uppsala, Sweden. Association for Computational Linguistics.

Specia, L., Turchi, M., Cancedda, N., Dymetman, M., and Cristianini, N. (2009a). Estimating the sentence-level quality of machine translation systems. In *Proceedings of the 13th Annual Conference of the European Association for Machine Translation*, pages 28–35, Barcelona, Spain.

Specia, L., Turchi, M., Wang, Z., Shawe-Taylor, J., and Saunders, C. (2009b). Improving the confidence of machine translation quality estimates. In *Proceedings of the Machine Translation Summit.*

Specia, L., Turqui, M., Wang, Z., Shawe-Taylor, J., and Saunders, C. (2009c). Improving the confidence of machine translation quality estimates. In *Proceedings of the Twelfth Machine Translation Summit (MT Summit XII)*. International Association for Machine Translation.

Steinberger, R., Pouliquen, B., Widiger, A., Ignat, C., Erjavec, T., Tufis, D., and Varga, D. (2006). The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. In *5th Edition of the International Conference on Language Resources and Evaluation (LREC '06)*, pages 2142–2147.

Ueffing, N. and Ney, H. (2005). Application of word-level confidence measures in interactive statistical machine translation. In *Proceedings of the European Association for Machine Translation conference*, pages 262–270.

Wold, H. (1966). *Estimation of Principal Components and Related Models by Iterative Least squares*, pages 391–420. Academic Press, New York.

Zhechev, V. and van Genabith, J. (2010). Seeding statistical machine translation with translation memory output through tree-based structural alignment. In *Proceedings of the 4th Workshop on Syntax and Structure in Statistical Translation*, pages 43–51, Beijing, China. Coling 2010 Organizing Committee.

Attachment A

Task 3.1

González-Rubio, J., Sanchís, A., and Casacuberta, F. (2012).

Prhlt submission to the wmt12 quality estimation task.

In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 104–108, Montréal, Canada. Association for Computational Linguistics.

PRHLT Submission to the WMT12 Quality Estimation Task

Jesús González Rubio and **Alberto Sanchis** and **Francisco Casacuberta**
D. Sistemas Informáticos y Computación
Universitat Politècnica de València
Camino de vera s/n, 46022, Valencia, Spain
{jegonzalez,josanna,fcn}@dsic.upv.es

Abstract

This is a description of the submissions made by the pattern recognition and human language technology group (PRHLT) of the Universitat Politècnica de València to the quality estimation task of the seventh workshop on statistical machine translation (WMT12). We focus on two different issues: how to effectively combine subsequence-level features into sentence-level features, and how to select the most adequate subset of features. Results showed that an adequate selection of a subset of highly discriminative features can improve efficiency and performance of the quality estimation system.

1 Introduction

Quality estimation (QE) (Ueffing et al., 2003; Blatz et al., 2004; Sanchis et al., 2007; Specia and Farzindar, 2010) is a topic of increasing interest in machine translation (MT). It aims at providing a quality indicator for unseen translations at various granularity levels. Different from MT evaluation, QE do not rely on reference translations and is generally addressed using machine learning techniques to predict quality scores.

Our main focus in this article is in the combination of subsequence features into sentence features, and in the selection of a subset of relevant features to improve performance and efficiency. Section 2 describes the features and the learning algorithm used in the experiments. Section 3 describe two different approaches implemented to select the best-performing subset of features. Section 4 displays the results of the experimentation intended to determine the optimal setup to train our final submission. Finally, section 5 summarizes the submission and discusses the results.

2 Features and Learning Algorithm

2.1 Available Sources of Information

The WMT12 QE task is carried out on English–Spanish news texts produced by a phrase-based MT system. As training data we are given 1832 translations manually annotated for quality in terms of post-editing effort (scores in the range $[1, 5]$), together with their source sentences, decoding information, reference translations, and post-edited translations. Additional training data can be used, as deemed appropriate. Any of these information sources can be used to extract the features, however, test data consists only on source sentence, translation, and search information. Thus, features were extracted from the sources of information available in test data only. Additionally, we compute some extra features from the WMT12 translation task (WMT12TT) training data.

2.2 Features

We extracted a total of 475 features classified into sentence-level and subsequence-level features. We considered subsequences of sizes one to four.

Sentence-level features

- Source and target sentence lengths, and ratio.
- Proportion of dead nodes in the search graph.
- Number of source phrases.
- Number and average size of the translation options under consideration during search.

- Source and target sentence probability and perplexities computed by language models of order one to five.
- Target sentence probability, probability divided by sentence length, and perplexities computed by language models of order one to five. Language models were trained on the 1000-best translations.
- 1000-best average sentence length, 1000-best vocabulary divided by average length, and 1000-best vocabulary divided by source sentence length.
- Percentage of subsequences (sizes one to four) previously unseen in the source training data.

Subsequence-level features

- Frequency of source subsequences in the WMT12TT data.
- IBM Model-1 confidence score for each word in the translation (Ueffing et al., 2003).
- Subsequence confidence scores computed on 1000-best translations as described in (Ueffing et al., 2003; Sanchis et al., 2007). We use four subsequence correctness criteria (Levensthein position, target position, average position, and any position) and three weighting schemes (translation probability, translation rank, and relative frequencies).
- Subsequence confidence scores computed by a smoothed naïve bayes classifier (Sanchis et al., 2007). We computed a confidence score for each correctness criteria (Levensthein, target, average and any). The smoothed classifier was tuned to improve classification error rate on a separate development set (union of news-test sets for years 2008 to 2011).

2.3 Combination of Subsequence-level Features

Since WMT12 focuses on sentence-level QE, subsequence-level features must be combined to obtain sentence-level indicators. We used two different methods to combine subsequence features:

- Average value of subsequence-level scores, as done in (Blatz et al., 2004).
- Percentage of subsequence scores belonging to each frequency quartile[1], as done in (Specia and Farzindar, 2010).

Thus, each subsequence-level feature was represented as five sentence-level features: one average score plus four quartile percentages.

Both methods aim at summarizing the scores of the subsequences in a translations. The average is a rough indicator that measures the “middle” value of the scores while the percentages of subsequences belonging to each quartile are more fine-grained indicators that try to capture how spread out the subsequence scores are.

2.4 Learning Algorithm

We trained our quality estimation model using an implementation of support vector machines (Vapnik, 1995) for regression. Specifically, we used SVM$^{\text{light}}$ (Joachims, 2002) for regression with a radial basis function kernel with the parameters C, w and γ optimized. The optimization was performed by cross-validation using ten random subsamples of the training set (1648 samples for training and 184 samples for validation).

3 Feature Selection

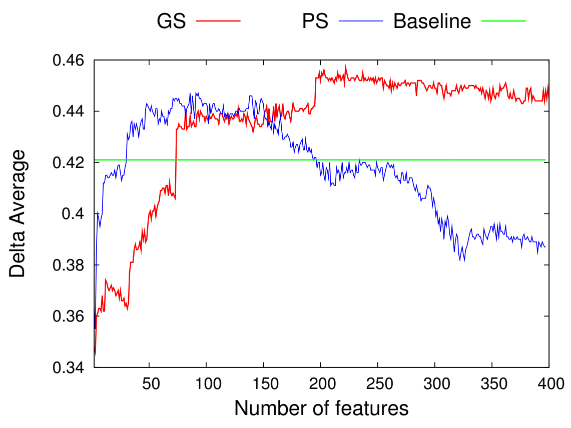
One of the principal challenges that we had to confront is the small size of the training data (only 1832 samples) in comparison with the large number of features, 475. This inadequate amount of training data did not allow for an acceptable training of the regression model which yielded instable systems with poor performance. We also verified that many features were highly correlated and were even redundant sometimes. Since the amount of training data is fixed, we tried to improve the robustness of our regression systems by selecting a subset of relevant features.

We implemented two different feature selection techniques: one based on partial component analysis (PCA), and a greedy selection according to the individual performance of each feature.

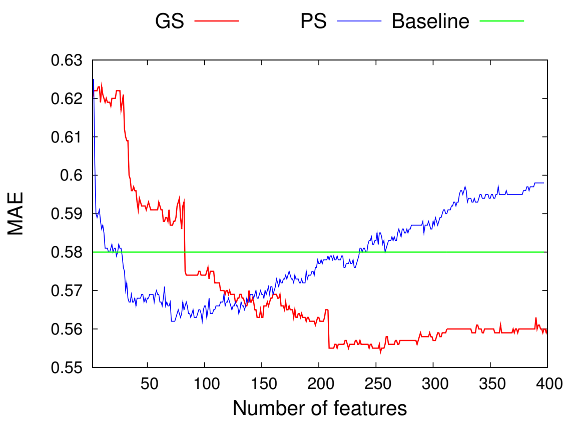
3.1 PCA Selection (PS)

Principal component analysis (Pearson, 1901) (PCA) is a mathematical procedure that uses an or-

[1]Quartile values were computed on the WMT12TT data.



(a) Delta average score



(b) Mean Average Error

Figure 1: Delta average score (a) (higher is better) and mean average error (b) (lower is better) as a function of the number of features. Cross-validation results for PCA selection (PS), and greedy selection (GS) methods.

thogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it be uncorrelated with the preceding components. Strictly speaking, PCA does not perform a feature selection because the principal components are linear combinations of the individual features.

PCA generates sets of features (the principal components) with almost no correlation. However, it ignores the quality scores to be predicted. Since we want to obtain the best-performing subset of features, there is a mismatch between the selection criterion of PCA and the criterion we are interested in. In other words, although the features generated by PCA contain almost no redundancy, they do not necessarily have to constitute the best-performing subset of features.

3.2 Greedy Performance-driven Selection (GS)

We also implemented a greedy feature selection method which iteratively creates subsets of increasing size with the best-scoring individual features. The score of each feature is given by the performance of a system trained solely on that feature. At a given iteration, we select the K best scoring features and train a regression system with them.

Since we select the features incrementally according to their individual performance, we expect to obtain the subset of features that yield the best performance. However, we do not take into account the correlations that may exist between the different features, thus, the final subset is almost sure to contain a large number of redundant features.

4 Experiments

4.1 Assessment Measures

The organizers propose two variations of the task that will be evaluated separately:

Ranking: Participants are required to submit a ranking of translations. This ranking will used to split the data into n quantiles. The evaluation will be performed in terms of delta average score, the average difference over n between the scores of the top quantiles and the overall score of the corpus. The Spearman correlation will be used as tie-breaking metric.

Scoring: Participants are required to assign a score in the range $[1, 5]$ for each translation. The evaluation will be performed in terms of mean average error (MAE). Root mean squared error (RMSE) will be used as tie-breaking metric.

4.2 Pre-Submission Results

We now describe a number of experiments whose goal is to determine the optimal training setup.

Specifically, we wanted to determine which selection method to use (PCA or greedy) and which features yield a better system. As a preliminary step, we extracted all the features described in section 2. The complete training data consisted on 1832 samples each one with 475 features.

We trained systems using feature sets of increasing size as given by PCA selection (PS) or greedy selection (GS). The parameters of each system were tuned to optimize each of the evaluation measures under consideration. Performance was measured as the average of a ten-fold cross-validation experiment on the training data.

Figure 1 shows the results obtained for the experiments that optimized delta average, and MAE (result optimizing Spearman and RMSE were quite similar). We also display the performance of a system trained on the baseline features. We observed that both selection methods yielded a better performance than the baseline system. PS allowed for a quick improvement in performance as more features are selected, reaching its best results when selecting approximately 80 features. After that, performance rapidly deteriorate. Regarding GS, its improvements in performance were slower in comparison with PS. However, GS finally reached the best scores of the experimentation when selecting $\sim$ 225 features. Specifically, the best performance was reached using the top 222 features for delta average, and using the top 254 features for MAE.

According to these results, our submissions were trained on the best subsets of features as given by the GS method. 222 features were selected according to their delta average score for the ranking task variation, and 254 according to their MAE value for the scoring task variation. Final submissions were trained on the complete training set.

Most of the selected features are sentence-level features calculated from subsequence-based scores. For instance, among the 222 features of the ranking variation of the task, 174 were computed from subsequence scores. Among these 174 features, 129 were calculated from confidence scores computed on 1000-best translations, 29 from confidence scores computed by a smoothed naïve bayes classifier, 11 from the frequencies of the subsequences in the WMT12TT data, and 5 from IBM Model-1 word confidence scores.

Participant ID	Delta average $\Uparrow$	MAE $\Downarrow$
SDL Language Weaver	0.63	0.61
Uppsala U.	0.58	0.64
LORIA Institute	–	0.68
Trinity College Dublin	0.56	0.68
Baseline	*0.55*	*0.69*
PRHLT	**0.55**	**0.70**
U. Edinburgh	0.54	0.68
Shanghai Jiao Tong U.	0.53	0.69
U. Wolverhampton/Sheffield	0.51	0.69
DFKI	0.46	0.82
Dublin City U.	0.44	0.75
U. Politècnica Catalunya	0.22	0.84

Table 1: Best official evaluation results on each task of the different participating teams. Results for our submissions are displayed in bold. Baseline results in italics.

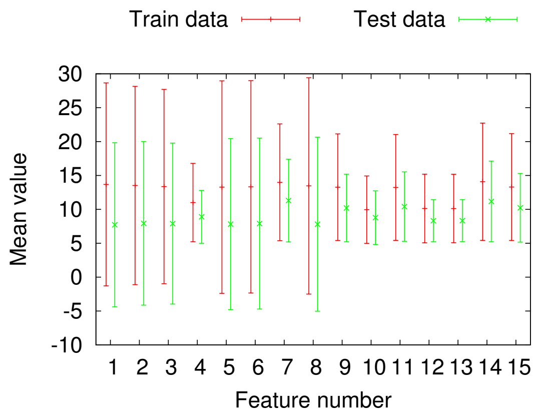


Figure 2: Average value ($\pm$ std. deviation) of the first 15 features used in our final submissions. Feature values follow a similar distribution in the training and test data.

4.3 Official Evaluation Results

After establishing the optimal training setup, we now show the official evaluation results for our submissions. Table 1 shows the performance of the various participants in the ranking (delta average) and scoring (MAE) tasks. Surprisingly our submissions yielded a slightly worse result than the baseline features. However, given the large improvements over the baseline system obtained in the pre-submission experiments, we expected to obtain similar improvements over Baseline in test.

We considered two possible explanations for this counterintuitive result. First, a possibly divergence between the underlying distributions of the training and test data. To investigate this possibility, we studied the distributions of feature values in the training and test data. Figure 2 displays mean$\pm$std. deviation for the first 15 features used in our final submissions (similar results are obtained for all the 222 features). We can observe that feature values in training and test data follow a similar distribution, although test values tend to be slightly lower than training values.

A second plausible explanation is the small amount of training data (only 1832 samples). Limited data favors simpler systems that can train its few free parameters more accurately. This is the case of the Baseline system that was trained using only 11 features, in comparison with the 222 features used in our submissions. Since the training and test data seem to have been generated following the same underlying distribution, we hypothesize that the limited training data is the main explanation for the poor test performance of our submissions.

5 Summary and Discussion

We have presented the submissions of the PRHLT group to the WMT12 QE task. The estimation systems were based on support vector machines for regression. Several features were used to train the systems in order to predict human-annotated post-editing effort scores. Our main focus in this article have been the combination of subsequence features into sentence features, and the selection of a subset of relevant features to improve the submitted systems performance.

Results of the experiments showed that PCA selection was able to obtain better performance when selecting a small number of features while GS yielded the best-performing systems but using much more features. Among the selected features, the larger percentage of them were calculated from subsequence features. These facts indicate that the combination of subsequence features yields sentence-level features with a strong individual performance. However, the high number of features selected by GS indicate that these top-scoring features are highly correlated.

Official evaluation results differ from what we expected; baseline system performs better than our submissions while pre-submission experiments yielded just opposite results. After discarding a possibly discrepancy between training and test data distributions, and given that smaller models such as the baseline system can be trained more accurately with limited data, we concluded that the limited training data is the main explanation for the disparity between our training and test results.

A future line of research could be the study of methods that allow to select sets of uncorrelated features, that unlike PCA, also take into account the individual performance of each feature. Specifically, we plan to study a features selection technique based on partial least squares regression.

Acknowledgments

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n^{o} 287576. Work also supported by the EC (FEDER/FSE) and the Spanish MEC under the MIPRCV "Consolider Ingenio 2010" program (CSD2007-00018) and iTrans2 (TIN2009-14511) project and by the Generalitat Valenciana under grant ALMPR (Prometeo/2009/01).

References

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. 2004. Confidence estimation for machine translation. In M. Rollins, editor, *Mental Imagery*. Yale University Press.

Thorsten Joachims. 2002. SVM light.

Karl Pearson. 1901. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(11):559–572.

Alberto Sanchis, Alfons Juan, and Enrique Vidal. 2007. Estimation of confidence measures for machine translation. In *In Procedings of the MT Summit XI*. Springer-Verlag.

Lucia Specia and Atefeh Farzindar. 2010. Estimating machine translation post-editing effort with hter. In *AMTA 2010- workshop, Bringing MT to the User: MT Research and the Translation Industry*. The Ninth Conference of the Association for Machine Translation in the Americas, nov.

Nicola Ueffing, Klaus Macherey, and Hermann Ney. 2003. Confidence measures for statistical machine translation. In *In Proceedings of the MT Summit IX*, pages 394–401. Springer-Verlag.

Vladimir N. Vapnik. 1995. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA.

Attachment B

Task 3.1

Buck, C. (2012).

Black box features for the wmt 2012 quality estimation shared task.

In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 91–95, Montréal, Canada. Association for Computational Linguistics.

Black Box Features for the WMT 2012 Quality Estimation Shared Task

Christian Buck
School of Informatics
University of Edinburgh
Edinburgh, UK, EH8 9AB
christian.buck@ed.ac.uk

Abstract

In this paper we introduce a number of new features for quality estimation in machine translation that were developed for the WMT 2012 quality estimation shared task. We find that very simple features such as indicators of certain characters are able to outperform complex features that aim to model the connection between two languages.

1 Introduction and Task

This paper describes the features and setup used in our submission to the WMT 2012 quality estimation (QE) shared task. Given a machine translation (MT) system and a corpus of its translations which have been rated by humans, the task is to build a predictor that can accurately estimate the quality of further translations. The human ratings range from 1 (incomprehensible) to 5 (perfect translation) and are given as the mean rating of three different judges.

Formally we are presented with a source sentence f_1^J and a translation e_1^I and we need to assign a score $S(f_1^J, e_1^I) \in [1, 5]$ or, in the ranking task, order the source-translation pairs by expected quality.

2 Resources

The organizers have made available a baseline QE system that consists of a number of well established features (Blatz et al., 2004) and serves as a starting point for development. Furthermore the MT system that generated the translations is available along with its training data. Compared to the large training corpus of the MT engine, the QE system is based on a much smaller training set as detailed in Table 1.

	# sentences
europarl-nc	1,714,385
train	1,832
test	422

Table 1: Corpus statistics

3 Features

In the literature (Blatz et al., 2004) a large number of features have been considered for confidence estimation. These can be grouped into four general categories:

1. *Source features* make a statement about the source sentence, assessing the difficulty of translating a particular sentence with the system at hand. Some sentences may be very easy to translate, e.g. short and common phrases, while long and complex sentences are still beyond the system's capabilities.

2. *Translation features* model the connection between source and target. While this is very closely related to the general problem of machine translation, the advantage in confidence estimation is that we can exercise unconstructive criticism, i.e. point out errors without offering a better translation. In addition, there is no need for an efficient search algorithm, thus allowing for more complex models.

3. *Target features* judge the translation of the system without regarding in which way it was produced. They often resemble the language

model used in the noisy channel formulation (Brown et al., 1993) but can also pinpoint more specific issues. In practice, the same features as for the source side can be used; the interpretation however is different.

4. *Engine features* are often referred to as *glass box features* (Specia et al., 2009). They describe the process which produced the translation in question and usually rely on the inner workings of the MT system. Examples include model scores and word posterior probabilities (WPP) (Ueffing et al., 2003).

In this work we focus on the first three categories and ignore the particular system that produced the translations. Such features are commonly referred to as *black box features*. While some glass box features, e.g. word posterior probabilities, have led to promising results in the past, we chose to explore new features potentially applicable to translations from any source, e.g. translations found on the web.

3.1 Binary Indicators

MTranslatability (Bernth and Gdaniec, 2001) gives a notion of the structural complexity of a sentence that relates to the quality of the produced translation. In the literature, several characteristics that may hinder proper translation have been identified, among them poor grammar and misplaced punctuation. As a very simple approximation we implement binary indicators that detect clauses by looking for quotation marks, hyphens, commas, etc. Another binary feature marks numbers and uppercase words.

3.2 Named Entities

Another aspect that might pose a potential problem to MT is the occurrence of words that were only observed a few times or in very particular contexts, as it is often the case for Named Entities. We used the Stanford NER Tagger (Finkel et al., 2005) to detect words that belong to one of four groups: Person, Location, Organization and Misc. Each group is represented by a binary feature.

Counts are given in Table 2. The test set has significantly less support for the *Misc* category, possibly hinting that this data was taken from a different source or document. To avoid the danger of biasing the classifier we decided not to use the *Misc* indicator in our experiments.

	train (src)		test (src)	
	abs	rel	abs	rel
Person	623	34%	141	33%
Location	479	26%	99	23%
Organization	505	28%	110	26%
Misc	428	23%	53	13%

Table 2: Distribution of Named Entities. The counts are based on a binary features, i.e. multiple occurrences are treated as a single one.

3.3 Backoff Behavior

In related work (Raybaud et al., 2011) the backoff behavior of a 3-gram LM was found to be the most powerful feature for word level QE. We compute for each word the longest seen n-gram (up to $n = 4$) and take the average length as a feature. N-grams at the beginning of a sentence are extended with `<s>` tokens to avoid penalizing short sentences. This is done on both the source and target side.

3.4 Discriminative Word Lexicon

Following the approach of Mauser et al. (2009) we train log-linear binary classifiers that directly model $p(e|f_1^J)$ for each word $e \in e_1^I$:

$$p(e|f_1^J) = \frac{exp\left(\sum_{f \in f_1^J} \lambda_{e,f}\right)}{1 + exp\left(\sum_{f \in f_1^J} \lambda_{e,f}\right)} \quad (1)$$

where $\lambda_{e,f}$ are the trained model weights. Please note that this introduces a global dependence on the source sentence so that every source word may influence the choice of all words in e_1^I as opposed to the local dependencies found in the underlying phrase-based MT system.

Assuming independence among the words in the translated sentence we could compute the probability of the sentence pair as:

$$p(e_1^I|f_1^J) = \prod_{e \in e_1^I} p(e|f_1^J) \cdot \prod_{e \notin e_1^I} \left(1 - p(e|f_1^J)\right). \quad (2)$$

In practice the second part of Equation (2) is too noisy to be useful given the large number of words

source	~~resumption~~ of the session
target	reanudación del período de ~~sesiones~~

Table 3: Example entry of filtered training corpus.

that do not appear in the sentence at hand. We therefore focus on the observed words and use the geometric mean of their individual probabilities:

$$x_{\text{DWL}}(f_1^J, e_1^I) = \left(\prod_{e \in e_1^I} p(e|f_1^J) \right)^{1/I} . \quad (3)$$

We also compute the probability of the lowest scoring word as an additional feature:

$$x_{\text{DWLmin}}(f_1^J, e_1^I) = \min_{e \in e_1^I} p(e|f_1^J). \quad (4)$$

3.5 Neural Networks

We seek to directly predict the words in e_1^I using a neural network. In order to do so, both source and target sentence are encoded as high dimensional vectors in which positive entries mark the occurrence of words. This representation is commonly referred to as the *vector space model* and has been successfully used for information retrieval.

The dimension of the vector representation is determined by the respective sizes of the source and target vocabulary. Without further pre-processing we would need to learn a mapping from a 90k ($|V_f|$) to a 170k ($|V_e|$) dimensional space. Even though our implementation is specifically tailored to exploit the sparsity of the data, such high dimensionality makes training prohibitively expensive.

Two approaches to reduce dimensionality are explored in this work. First, we simply remove all words that never occur in the QE data of 2,254 sentences from the corpus leaving 8,365 input and 9,000 output nodes. This reduces the estimated training time from 11 days to less than 6 hours per iteration[1]. Standard stochastic gradient decent on a three-layer feed-forward network is used.

As shown in Table 3 the filtering can lead to artifacts in which case an erroneous mapping is learned. Moreover the filtering approach does not scale well as the QE corpus and thereby the vocabulary grows.

[1]using a 2.66 GHz Intel Xeon and 2 threads

Our second approach to reduce dimensionality uses the *hashing trick* (Weinberger et al., 2009): a hash function is applied to each word and the sentence is represented by the hashed values which are again transformed using vector space model as above. The dimensionality reduction is due to the fact that there are less possible hash values than words in the vocabulary. To reduce the loss of information due to collisions, several different hash functions are used. The resulting vector representation closely resembles a Bloom Filter (Bloom, 1970).

This approach scales well but introduces two new parameters: the number of hash functions to use and the dimensionality of the resulting space. In our experiments we have used SHA-1 hashes with three different salts of which we used the first 12 bits, thereby mapping the sentences into a 4096-dimensional space.

The results presented in Section 4 based on networks with 500 hidden nodes which were trained for at least 10 iterations. The networks are not trained until convergence due to time constraints; additional training iterations will likely result in better performance. Experiments using 250 or 1000 hidden nodes showed very similar results.

After the models are trained we compare the predicted and the observed target vectors and derive two features: (i) the euclidean distance, denoted as NNdist and HNNdist for the filtered and hashed versions respectively and (ii) the geometric mean of those dimensions where we expect a positive value, denoted as NNprop+ and HNNprob+ in Table 5.

3.6 Edit Distance

Using Levenshtein Distance we computed the distance to the closest entry in the training corpus. The idea is that a sentence that was already seen almost identically would be easier to translate. Likewise, a translation that is very close to an element of the corpus is likely to be a good translation. This was performed for both source and target side and on character as well as on word level giving a total of four (EDIT) scores. The scores are normalized by the length of the respective lines.

source corpus	“	”	"
europarl-nc	37	227	25,637
train	0	0	641
test	78	76	100

Table 4: Counts of different quotation mark characters.

4 Experiments

In this work we focus on the prediction of human assessment of translation quality, i.e. the regression task of the WMT12 QE shared task. Our submission for the ranking task is derived from the order implied by the predicted scores without further re-ranking.

In general our efforts were directed towards feature engineering and not to the machine learning aspects. Therefore, we apply a standard pipeline and use neural networks for regression. All parameter tuning is performed using 5-fold cross validation on the baseline set of 17 features as provided by the organizers.

4.1 Preprocessing and Analysis

To avoid including our own judgment, no more than the first ten lines of the test data were visually inspected in order to ensure that the training and test data was preprocessed in the same manner. Furthermore, the distribution of individual characters was investigated. As shown in Table 4, the test data differs from the training corpus in treatment of quotation marks. Hence, we replaced all typographical quotation marks (“, ”) with the standard double quote symbol (").

Prior to computation of the features described in Subsections 3.3, 3.4 and 3.5 all numbers are replaced with a special `$number` token.

Baseline features are used without further scaling; experiments where all features were scaled to the $[0, 1]$ range showed a drop in accuracy.

While we implemented the training ourselves for the features presented in Subsection 3.5, the open source neural network library FANN[2] is used for all experiments in this section. As the performance of individual classifiers shows a high variance, presumably due to local minima, all experiments are conducted using ensembles on 500 networks trained with random initialization. Their consensus is computed as the average of the individual predictions.

[2]http://leenissen.dk/fann/wp/

Feature (Section)	MAE	RMSE	|PCC|
BACKOFF (3.3)	0.0	0.0	
INDICATORS (3.1)	+0.5	+0.7	
NER (3.2)	+0.5	+0.4	
DWLmin (3.4)	−0.1	−0.1	0.19
DWL (3.4)	0.0	−0.1	0.36
EDIT (3.6) - tgt words	0.0	0.0	0.32
EDIT (3.6) - tgt chars	−0.1	0.0	0.27
EDIT (3.6) - src words	0.0	0.0	0.36
EDIT (3.6) - src chars	+0.2	+0.1	0.37
NNdist (3.5)	0.0	0.0	0.35
NNprob+ (3.5)	+0.1	+0.2	0.35
HNNdist (3.5)	0.0	0.0	0.37
HNNprob+ (3.5)	+0.1	+0.1	0.35

Table 5: Analysis of individual features using 5-fold cross-validation. Positive values indicate improvement over a baseline of MAE 57.7% and RMSE 72.7%; e.g. including the DWL feature actually worsens RMSE from 72.7% to 72.8%.
The last column gives the Pearson correlation coefficient between the feature and the score if the feature is a single column. This information was not used in feature selection as it is not based on cross validation.

4.2 Feature Evaluation

To evaluate the contribution of individual features, each feature is tested in conjunction with all baseline features, using the parameters that were optimized on the baseline set. This slightly favors the baseline features but we still expect that expressive additional features lead to a noticeable performance gain. The results are detailed in Table 5. In addition to the main evaluation metrics, mean average error (MAE) and root mean squared error (RMSE), we report the Pearson correlation coefficient (PCC) as a measure of predictive strength of a single feature. Because features are not used alone this does not directly translate into overall performance. Still, it can be observed that our proposed features show good correlation to the target variable. For comparison, among the baseline features only 2 of 17 reach a PCC of over 0.3.

While the results generally remain inconclusive, some very simple features that indicate difficulties for the translation engine show good performance. In particular binary markers of named entities and and the indicator features introduced in Subsection 3.1 perform well. Further experiments with the latter show their contribution to the systems performance can be attributed to a single feature: the indicator of the genitive case, i.e. occurrences of **'s** or **s'**.

Testing more combinations of simple and complex features may lead to improvements at the risk of over-fitting on the cross validation setup. As a simple remedy several feature sets were created at random, always combining all baseline features and several new features presented in this paper. Averaging of the individual results of all sets that performed better than the baseline resulted in our submission.

4.3 Results and Discussion

Of all the features detailed only a few lead to a considerable improvement. This is also reflected by our results on the test data which are nearly indistinguishable from the performance of the baseline system. While this is disappointing, our more complex features introduce a number of free parameters and further experimentation will be needed to conclusively assess their usefulness. In particular, features based on neural networks can be further optimized and tested in other settings.

Even though the machine learning aspects of this task are not the focus of this work we are confident that the proposed setup is sound and can be reused in further evaluations.

5 Conclusion

We described a number of new features that can be used to predict human judgment of translation quality. Results suggest pointing out sentences that are hard to translate, e.g. because they are too complex, is a promising approach.

We presented a detailed evaluation of the utility of individual features and a solid baseline setup for further experimentation. The system, based on an ensemble of neural networks, is insensitive to parameter settings and yields competitive results.

Our new features can potentially be applied for a multitude of applications and may deliver insights into the fundamental problems that cause translation errors, thus aiding the progress in MT research.

Acknowledgments

This work was supported by the MateCAT project, which is funded by the EC under the 7^{th} Framework Programme.

References

Arendse Bernth and Claudia Gdaniec. 2001. Mtranslatability. *Machine Translation*, 16(3):175–218, September.

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. 2004. Confidence estimation for machine translation. In *Proceedings of Coling 2004*, pages 315–321, Geneva, Switzerland, August.

Burton H. Bloom. 1970. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, July.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–312.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL 2005, pages 363–370, Stroudsburg, PA, USA.

Arne Mauser, Saša Hasan, and Hermann Ney. 2009. Extending statistical machine translation with discriminative and trigger-based lexicon models. In *Conference on Empirical Methods in Natural Language Processing*, pages 210–217, Singapore, August.

Sylvain Raybaud, David Langlois, and Kamel Smaïli. 2011. "this sentence is wrong." detecting errors in machine-translated sentences. *Machine Translation*, 25(1):1–34, March.

Lucia Specia, Nicola Cancedda, Marc Dymetman, Marco Turchi, and Nello Cristianini. 2009. Estimating the sentence-level quality of machine translation systems. In *Proceedings of the 13th Annual Conference of the European Association for Machine Translation*, EAMT-2009, pages 28–35, Barcelona, Spain, May.

Nicola Ueffing, Klaus Macherey, and Hermann Ney. 2003. Confidence measures for statistical machine translation. In *Machine Translation Summit*, pages 394–401, New Orleans, LA, September.

Kilian Q. Weinberger, Anirban Dasgupta, John Langford, Alexander J. Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning. In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, ACM International Conference Proceeding Series, pages 1113–1120, Montreal, Quebec, Canada, June.